University of Toronto

CSCB28F—File Structures, Fall 1997

Assignment 3: On line processing: B-trees

Due: Wednesday 3 December; electronic submission at 10:00 am, and report at 11:10 am.
Partners: you may complete and submit this assignment with a single partner from the class. Both partners will receive the same mark.
Worth: 15%.

Introduction

The purpose of this assignment is to teach you B-trees and direct access using hashing with overflow and chaining. To learn about these important techniques, you should read your lecture notes on the topic and Chapters 8, 9 and 10 in the textbook.

You are to write programs that maintain a set of flights and a set of passengers. The users will make requests of the following types: add a new flight; delete a flight; add a passenger to a flight; and delete a passenger from a flight. The information is maintained in two file: *flights* (hashing with chained overflow indexed by *flight-id*), and *passengers* (B+-tree indexed by the pair [*flight-id*, *passenger-name*]). Your task is to process all transactions (add/delete flights, and add/delete passengers) contained in a transaction file updating both *flights* and *passengers* files. When deleting a flight, your program must delete all passengers in that flight (that's why you need a B+-tree). The passengers file records must contain passenger-name and flight-id; other fields should be added by you. The flight file description could be as in previous assignments.

Your program and report

For this assignment, you must provide a report with your submission. It must include: field description for the records on each file involved in your program; clear explanation on how each of the four type of transactions was implemented and how one file is accessed from another file; thorough tests stating what your program accomplish and what does not accomplish; source listings; etc. In addition, the report must explicitly state who did what in your project (only for groups of two partners).

There may be errors in the transaction file and your program should do some error checking. Do not apply the transactions with errors to the flights and passengers files. Instead, just output the transaction and a short message to an error file. Your program can consider only errors that depend on the present content of the files (for instance, a deletion of a passenger from a flight s/he was not taking). Do not read the entire files all at once into main memory. In real situations, these files may be too large to fit in main memory.

How to hand in your assignment

You must submit **all** your programs and data files. Use the submit command to submit copies of your files.

Marks will be given for good style and modularity. The program should be properly documented with features such as comments for each variable and function, variables with sensible names, commented tests in the output files, and so on. The marker must be able to compile your program using **gcc**, and run it successfully on an appropriate test file.